XT

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/828,049	04/06/2001	Jason Souloglou		5766

36183 7590 08/26/2004

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	PAPER NUMBER
2122	8

DATE MAILED: 08/26/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/828,049	SOULOGLOU ET AL.
	Examiner	Art Unit
	Chih-Ching Chow	2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04/06/2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *06 April 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☒ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *08/24/2001*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

Priority

1. It is noted that this application appears to claim subject matter disclosed in prior Application No. PCT/GB99/03168, filed 11 October 1999, and prior Application No. (GB) 9822075.9, filed 10 October 1998. A reference to the prior application must be inserted as the first sentence of the specification of this application or in an application data sheet (37 CFR 1.76), if applicant intends to rely on the filing date of the prior application under 35 U.S.C. 119(e) or 120. See 37 CFR 1.78(a). For benefit claims under 35 U.S.C. 120, the reference must include the relationship (i.e., continuation, divisional, or continuation-in-part) of all nonprovisional applications. Also, the current status of all nonprovisional parent applications referenced should be included.

2. Acknowledgment is made of applicant's claim for foreign priority based on an application filed in Application No. PCT/GB99/03168 on 11 October 1999 and a prior Application No. (GB) 9822075.9, filed 10 October 1998. It is noted, however, that applicant has not filed a certified copy of either the Application No. PCT/GB99/03168 application nor the Application No. (GB) 9822075.9 application as required by 35 U.S.C. 119(b).

3. Since this application is a CIP (Continuation-In-Part), for priority purpose, the priority date is 06 April, 2001.

Drawings

4. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(4) because reference characters 1, 2 (FIG 1), 3 have been used to designate FIGs. 1-5 (different parts with same number). Corrected drawing sheets are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

5. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description: characters 1-12 in FIGs 1-5. Corrected drawing sheets, or amendment to the specification to add the reference character(s) in the description, are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR

1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

6. The following guidelines illustrate the preferred layout for the specification of a utility application. These guidelines are suggested for the applicant's use.

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

(a) TITLE OF THE INVENTION.
(b) CROSS-REFERENCE TO RELATED APPLICATIONS.
(c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.
(d) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC (See 37 CFR 1.52(e)(5) and MPEP 608.05. Computer program listings (37 CFR 1.96(c)), "Sequence Listings" (37 CFR 1.821(c)), and tables having more than 50 pages of text are permitted to be submitted on compact discs.) or
REFERENCE TO A "MICROFICHE APPENDIX" (See MPEP § 608.05(a). "Microfiche Appendices" were accepted by the Office until March 1, 2001.)
(e) BACKGROUND OF THE INVENTION.
(1) Field of the Invention.
(2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.
(f) BRIEF SUMMARY OF THE INVENTION.
(g) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).
(h) DETAILED DESCRIPTION OF THE INVENTION.
(i) CLAIM OR CLAIMS (commencing on a separate sheet).

(j) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).
(k) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

Double Patenting

7. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefore ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

8. Claims 1-11 are provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-11 of copending Application No. 10/165,012. This is a

provisional double patenting rejection since the conflicting claims have not in fact been patented.

9. Claims 1-11 are provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-11 of copending Application No. 10/165,029. This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

10. Claims 1 item (i) and (ii) are provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 11 item (i) and (ii) of copending Application No. 09/827,974. This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 112

11. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> **The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.**

12. Claim 7 is rejected, the meaning of "The method of claim 6, wherein the optimizing step is used to optimize the program code written for execution by a processor of a first pipe..." is indefinite in the paragraph, should it be a 'type' instead of 'pipe'? An editorial review is required for this claim.

Claim Rejections - 35 USC § 103

13. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> **(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.**

14. Claims 1-4, 6-11 are rejected under 35 U.S.C. 103(a) as being unpatentable over Aho, Alfred et al. (hereinafter "Aho") "Compilers, principles, techniques, and tools" book as applied to claims above, and further in view of Davidson U.S. Pat. No. 5,613,117.

CLAIMS	**Aho / Davidson**
1. A method for generating an intermediate representation of program code written for running on programmable machine, said method comprising:	For Claim 1, first paragraph, Aho discloses the concept of generating **intermediate representation** of program code in his book, on page 12, section 'Intermediate Code Generation': "After syntax and semantic analysis, some computers generate an explicit **intermediate representation** of the source program. We can think of this **intermediate representation** as a program for an abstract machine."
(i) generating a plurality of register objects holding variable values be generated by the program code; and (ii) generating a plurality expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;	With respect to paragraph (i) and (ii). Aho teaches all aspects of the applicant's claims but it does not specifically mentioned "register objects holding variable values". However, Davidson shows an apparatus allows a plurality of register objects holding variable values. In Davidson, column 2, lines 39-44, "Next in the internal organization of a compiler is the register and memory allocation. Up to this point, data references were to variables and constants by name or in the abstract, without regard to where stored;

now, however, data references are assigned to more concrete locations, such as specific **registers** and memory displacements." Further in column 33, lines 41-47, "A data access tuple is a tuple which causes a **value** to be loaded from or stored into memory. (The word "memory" here includes **registers** in a register set of the target CPU 25. The only difference between a register and a normal memory location of the CPU 25 is that the 'address' of the register can only be used in a data access tuple.)" These two paragraphs taught us that the tuples may serve as register objects. The registers are inherited from Microprocessors, in order for Aho and Davidson to use them, registers have to be generated at some point. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **register allocation** of Aho with the register holding values further taught by Davidson, for the purpose of allowing the code generator to generate reasonable code for all target machines (See Davidson, column 35, lines 52-53).

For item (ii), b. an **expression object** is an <u>operation (operator)</u> performed for the register objects. The **expression objects** should be generated when the parsing of the program code is done. On Aho's book, page 49, under 'Abstract and Concrete Syntax' section, "A useful starting point for thinking about the translation of an input string is an *abstract syntax tree* in which each node represents an operator and the children of the node represent the operands." Here the <u>node</u> can be considered as a **register object** representing a respective variables and the operator is an **expression object** that is referenced by a **register object** (operand). An example is given in Aho's book, page 558-559, each of the t2, t3, t1 and t4 are 'expression objects' and the tree shows the 'relationship' in between the expression objects, see Aho, page 290, 'Directed Acyclic Graphs for Expressions', "A directed acyclic graph for an expression identifies the common subexpressions in the expression, therefore it shows the relationship in between the expressions. Davidson further discloses this point, in Davidson, column 7, lines 43-49, "The expression may also be expressed as a logic tree as seen in FIG. 3, where the tuples are identified by the same reference numerals. This same line of source code could be expressed in assembly for a RISC type target machine, as three instructions LOAD, ADD integer, and STORE, using some register such as REG4 in the register file, in the general form seen in FIG. 3." This paragraph shows generating the intermediate representation in terms of tuples that serve as expression objects. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **register allocation** of Aho with the expression registers holding fixed values or relationships between the objects further taught by Davidson, for the purpose of allowing the code generator to generate reasonable code for all target machines (See Davidson, column 35, lines 52-53).

(iii) wherein at least one variable sized register is represented by plural register objects, one register object being provided each possible size variably sized register.

For item (iii), Aho teaches all aspects of the applicant's claims but it does not specifically mention "variable sized register is represented by plural register objects", however Davidson discloses the concept of using variable sized register. In column 72, lines 22-25, "ALLOCATE.sub.--PERMANENT(operand, **size**) causes a permanent class TN (Temporary Name)

	of 'size' bytes to be created and referenced by the specified "operand" variable. If the "size" parameter is missing then the size of the TN is determined by the result data type of the current template." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **register allocation** of Aho with the flexible sized register allocation further taught by Davidson, for the purpose of allowing the code generator sufficient flexibility to generate reasonable code for all target machines (See Davidson, column 35, lines 52-53).
2. A method according to claim 1, wherein a write operation to a variably sized register is effected by writing to the register object corresponding to the appropriate size and maintaining a record of which register objects contain valid data.	For the feature of claims 1 see Claim 1 rejection. On Aho, page 537, under 'Register and Address Descriptors', "The code generation algorithm uses descriptors to keep track of register contents and addresses for names. 1. A register descriptor keeps track of what is currently in each register. 2. An address descriptor keeps track of the location (or locations) where the current value of the name can be found at run time. The location might be a register, a stack location, a memory address, or some set of these, since when copied, a value also stays where it was. This information can be stored in the symbol table and is used to determine the accessing methods for a name." The address descriptor **keeps track of the valid data** that are stored in the register objects as described in the claim, "maintaining a record of which register objects contain valid data". Aho teaches all aspects of the applicant's claims but it does not specifically mention "writing to the register object", however Davidson discloses the concept of **reading** from

	and **writing** to a variable sized register. In Davidson, column 31, lines 45-50, "ALLOCATE.sub.–symbol representing a variable which is allocated to a register does not have an address, in the normal sense. However, such a symbol may be used as the address operand of a tuple which reads from or writes to memory (a FETCH or STORE), in which case the tuple will access the indicated register." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **register descriptor** of Aho with the reading and writing operations further taught by Davidson, for the purpose of allowing value to be **loaded from** or **stored into** memory. See Davidson, column 33, lines 41-47.
3. A method according to claim 2, wherein a read operation from a variably sized register is effected by determining from said record if there is valid data in more than one corresponding register object which must be combined to give the same effect as reading from the variable register, and	For the feature of claims 2, see Claim 2 rejection. With regard to the 'read operation from a variably sized register', please see the rejection for claim 2 (Davidson).
(i) if it is determined that such combination is required, reading from the appropriate register object; and (ii) if it is determined that such combination is required, combining the contents appropriate register objects to provide a read value.	For item (i) and (ii), Aho discloses the skill to use multiregister for operations. See Aho, page 565, 'Multiregister Operations', "We can modify our labeling algorithm to handle operations like multiplication, division, or a function call, which normally require more than one register to perform. Simply modify step (6) of Fig. 9.23, the labeling algorithm, so label (n) is always at least the number of the registers required by the operation." The function 'label' is able to return the number of the registers required by the operation. In addition to the 'Multiregister Operation', Aho also mentioned 'register pair' concept, page 517, under 'Register Allocation', 5th

paragraph, "Certain machines require register-pairs (an even and next odd-numbered register) for some operands and results." Basically, registers can be defined in 8, 16, 32, 64 bits, it can always come in multiples. Therefore it's able to cover the function of combining many appropriate register objects needed for a certain read value. It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **Multiregister operation** of Aho with the reading and writing operations further taught by Davidson, for the purpose of allowing value to be **loaded from** or **stored into** multiple registers.

4. The method of claim 1, comprising translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the generated intermediate representation.

For the feature of claim 1 see rejection of claim 1. On Aho, page 463, first paragraph, "In the analysis-synthesis model of a compiler, the front end **translates** a source program into an **intermediate representation** form which the back end generates target code. Although a source program can be translated directly into the target language, some benefits of using a machine-independent intermediate form are:
1. Retargeting is facilitated; a compiler for a different machine can be created by attaching a back end of the new machine to an existing front end.
2. A machine-independent code optimizer can be done to the intermediate representation." Aho taught us that the program code from a front end processor (processor of a first type) can be translated into **intermediate representation** and the **intermediate representation can be optimized**, and then **be executed to an back end processor (processor of a second type)**. In addition, at the beginning of the current invention spec, 2nd

	paragraph, the inventor also mentioned that "**Intermediate representation** is a term <u>widely used in the computer industry</u> to refer to terms of abstract computer language in which a program may be expressed, but which is not specific to, and is not intended to be directly executed on, any particular processor...". Since the 'intermediate representation' concept has been 'widely used' therefore it's not an invention.
6. The method of claim 1, comprising optimizing the program code by optimizing said generated intermediate representation.	For the feature of claim 1 see rejection of claim 1. On Aho page 463, 3rd paragraph, "A machine-independent **code optimizer** can be applied to the **intermediate representation**."
7. The method of claim 6, wherein the optimizing step is used to optimize the program code written for execution by a processor of a first pipe so that the program code may be executed more efficiently by that processor.	For the feature of claim 6, see rejection of claim 6. Since the intermediate representation, which was generated from the program code that was written for execution by a processor (assuming it's first 'type' instead of 'pipe'), can be optimized, therefore the program code may be executed more efficiently by that processor.
8. A method of generating an intermediate representation of program code expressed in terms of the instruction set of a subject processor comprising at least one variable sized register, the method comprising the computer implemented steps (i) generating a set of associated abstract register objects representing the variable sized register;	Aho discloses the concept of generating **intermediate representation** of program code in his book, see rejection for claim 1. Aho also discloses the concept of "generating an intermediate representation of program code expressed in terms of the instruction set", in Aho's book, page 14, "we consider an intermediate form called 'three-address code,' which is like the assembly language for a machine in which every memory location can act like a register. Three-address code consists of **a sequence of instructions**, each of which has at most three operands." With respect to item (i), see rejection for

	claim 1 (iii). The term of abstract register Objects is used since they can hold any type of data (abstract data type).
(ii) for each write operation of a certain field width to the variable sized register, writing an abstract register of the same width;	With respect to item (ii), see the rejections of claim 1 (iii) and claim 2.
(iii) maintaining a record of which abstract register objects contain valid data, which record is updated upon each write operation; and	With respect to item (iii), see rejection of claim 2.
(iv) for each read operation a given field width, determining from said record whether there is valid data in more than one of said different sized abstract registers of the set which must be combined to give the same effect as the same read operation performed upon the variable size register; and (a) if it is determined that no combination is so required, reading directly from the appropriate register; or (b) if it is determined that data from more than one register must be so combined, combining the contents of those registers.	With respect to item (iv), see the rejections of claim 1 (iii), and claim 3; for items (a) and (b), see rejection 3 (i) and (ii).
9. The method according to claim 4, wherein the step of determining whether or not the contents of more than one abstract register must be combined and if so which abstract registers must be combined, is determined in accordance with following conditions respect each set of different sized abstract registers: (i) if the data required for an access lies wholly within one valid abstract register, that register only is accessed; and (ii) if the data required for an access lies within more than one valid abstract register, data is combined from those valid abstract registers to perform the access.	For the feature of claim 4, see rejection of claim 4. For the rest of the claim, see the rejection of claims 1, 3, and 8. With respect to item (i) and (ii) see rejection of claim 1 (i) and (ii), where recited that "A data access tuple is a tuple which causes a **value** to be loaded from or stored into memory. (The word "memory" here includes **regist**)". Only the required size is allocated, if data lies wholly within one valid abstract register, that register only is accessed. If the data required for an access lies within more than one register, that much amount of register space would be allocated (i.e. multiple

	registers may be combined).
10. A system for generating intermediate representation program code written for running on a programmable machine, the system comprising: (i) means for generating a plurality of register objects for holding variable values to be generated by the program code; and (ii) means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code; (iii) wherein at least one variably sized register is represented by plural register objects, one register object being provided for each possible size of the variably sized register.	See rejection of claim 1.
11. A system for generating an intermediate representation of program code expressed in terms of the instruction set of a subject processor comprising of at least one variably sized register, the system comprising: (i) means for generating set of associated abstract register objects representing the variably sized register; (ii) means for writing, for each write operation of a certain field width to the variable sized register to an abstract register object of the same width; (iii) means for maintaining a record of which abstract register objects contain valid data, the record being updated upon each write operation; and (vi) means for determining from said record, for each read operation of a given	For claim 11, see rejection of claim 8. For item (a) and (b) see rejections of claim 3 (i) Multiregister Operation, and claim 9 (i) and (ii).

width, whether there is valid data in more than one said different sized abstract registers of the set which must be combined to give the same effect as the same read operation performed upon the variable size register, and

(a) if is determined that no combination is so required, reading directly from the appropriate register; or

(b) if it is determined that data from more than one register must be so combined, combining the contents of those registers.

15. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Aho, Alfred et al. (hereinafter "Aho") "Compilers, principles, techniques, and tools" book as applied to claims above, further in view of Davidson U.S. Pat. No. 5,613,117, and further in view of U.S. Patent No. 6,463,582 by Lethin.

CLAIM	**Aho / Davidson / Lethin**
5. The method of claim 4, wherein the translation is performed dynamically as the program is run.	For the feature of claims 4 see claim 4 rejection which includes Aho and Davidson's disclosures. On Aho, page 347, under 'Static and Dynamic Checking of Types' section, "Checking done by a compiler is said to be static, while checking done when the target program runs is termed **dynamic**." Aho teaches all aspects of the applicant's claims but it does not specifically mention that the translation step is performed as the program code is run. However, Lethin shows the concept of translating step is performed dynamically as the program code is run. In Lethin's abstract, line 1, "An optimizing object code translation system and method perform **dynamic**

system and method perform **dynamic** compilation and translation of a target object code on a source operating system while performing optimization. **Compilation and optimization of the target code is dynamically executed in real time**." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the dynamic checking of Aho and Davidson, with the performing translation dynamically as the program code is run further taught by Lethin, for the purpose of improve the emulation and interpretation ability (see Lethin abstract line 6-7).

Conclusion

The following summarizes the status of all the claims:

103 rejections: 1 – 11

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 703-305-7205. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q Dam can be reached on 703-305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-308-3988.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Examiner
Art Unit 2122

CC

[illegible]

JOHN CHAVIS
PATENT EXAMINER
ART UNIT 2124